AMENDMENTS TO THE SPECIFICATION

Please amend the specification as follows *(wherein additions are shown by underlining and deletions are shown by strikethrough)*:

Please amend the paragraph beginning at page 22, line 3 as follows:

With the information from the drivers, the hibernation engine 200 can match the requirements and limitations of the hard disk controller 140 and DMA controller ~~206~~ 204 so that DMA hibernation is possible without requiring complex kernel services during the I/O. In other words, as long as the drivers' allocated data locations are appropriately sized in a safe memory range, along with a properly aligned output buffer 218, and that the hibernation engine 200 splits any large I/O requests into sequence of smaller ones that otherwise match the hardware limitations, DMA requests will not cause any additional operations with the I/O buffer or I/O request that may violate the memory-related rules of hibernation. Note that instead of using generic memory, the DMA-based I/O functions use the hibernation-safe memory location supplied via the pointer from the calling hibernation engine 200. Since there is no need for the use of complex kernel services for dequeuing pending I/O requests, no dequeuing is performed.

Please amend the paragraph beginning at page 22, line 20 as follows:

The hibernation engine starts the I/O operation by calling the DMA driver 210, which in turn instructs the DMA controller ~~206~~ 204 to start writing to the hard disk controller 140, and instructs the hard disk controller 140 to write a data sequence from the DMA controller ~~206~~ 204 to a specific location on the hard disk 141. The driver works by getting a new I/O (IO_START) request, including parameters for the new I/O request, including the size of the requested I/O operation, the target location on the hard disk, the address of the beginning of the I/O memory block, along with an address in the hibernation-safe memory region 202 for internal driver purposes. For safety reasons, the driver function verifies that each I/O request is compliant with the requirements imposed by the hardware it handles. If compliant, the function passes the request directly to lower-level routines without any transformations. If a request does not comply, the request is rejected and the driver returns STATUS_INVALID_IO_REQUEST.